* Data structure used:

→ ~~Linked~~ Linked List:-

• A linked list is an ordered collection of data in which each element contains minimum two values, data and link(s) to its successor (and/or) predecessor.

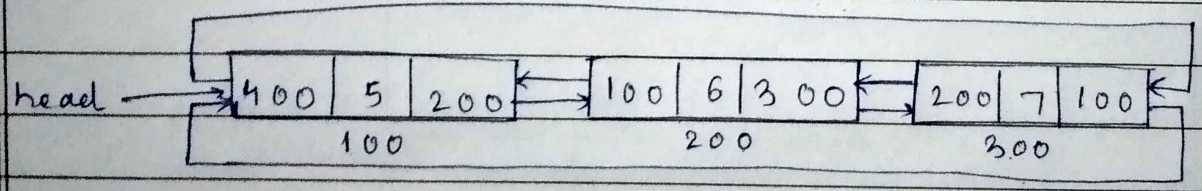
→ Type of linked list used:-

• Doubly circular linked list

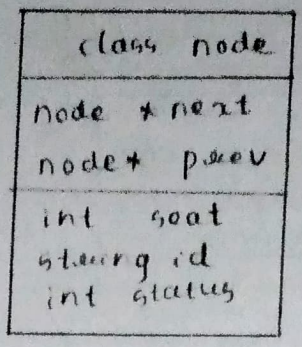
i) A circular doubly linked list is a mixture of a doubly linked list and a circular ~~to~~ linked list.

ii) Like the doubly linked list, it has an extra pointer called the previous pointer, and similar to the ~~it~~ circular linked list, its last node points to the head node.

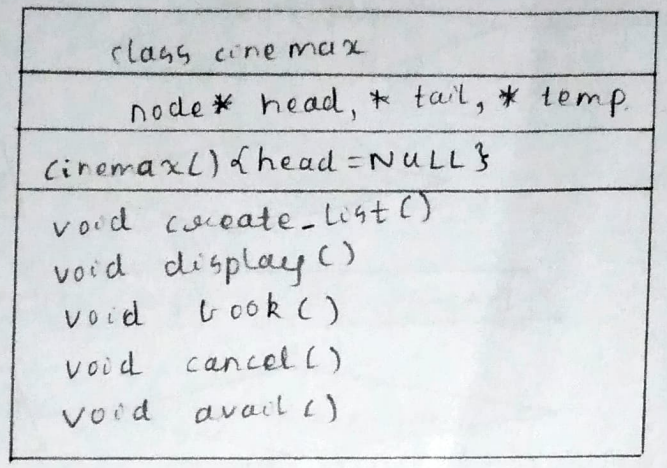
eg:



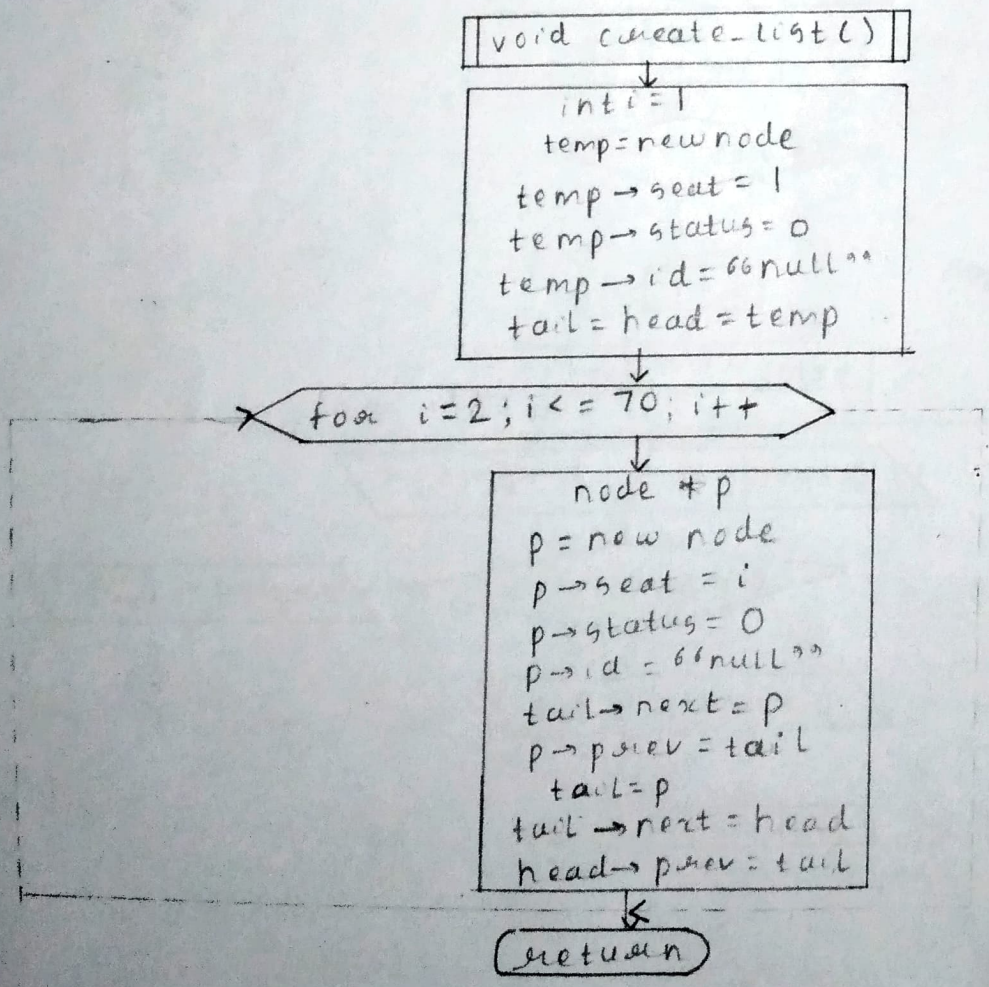
→ Flowchart for class node



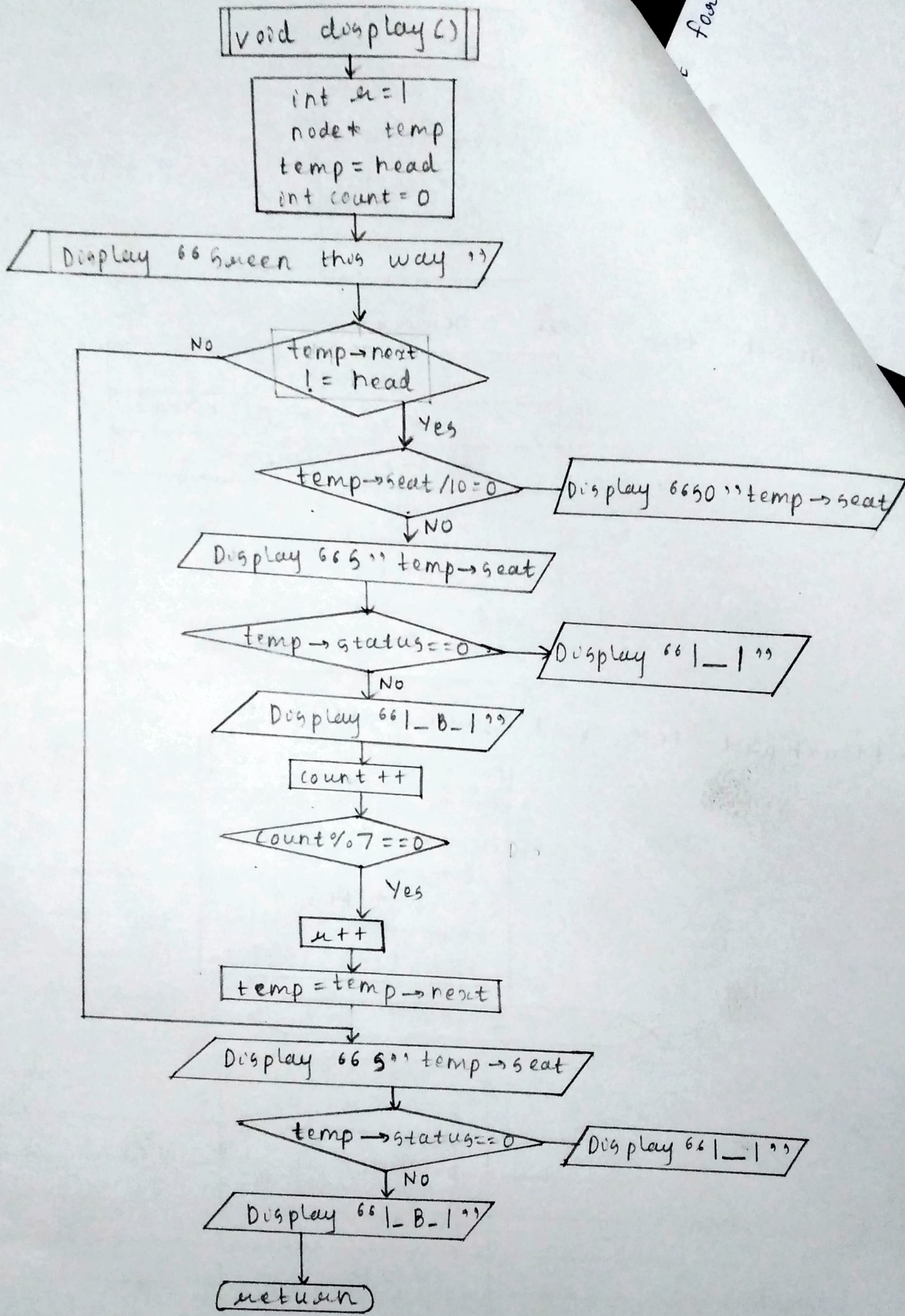
→ Flowchart for class cinemax



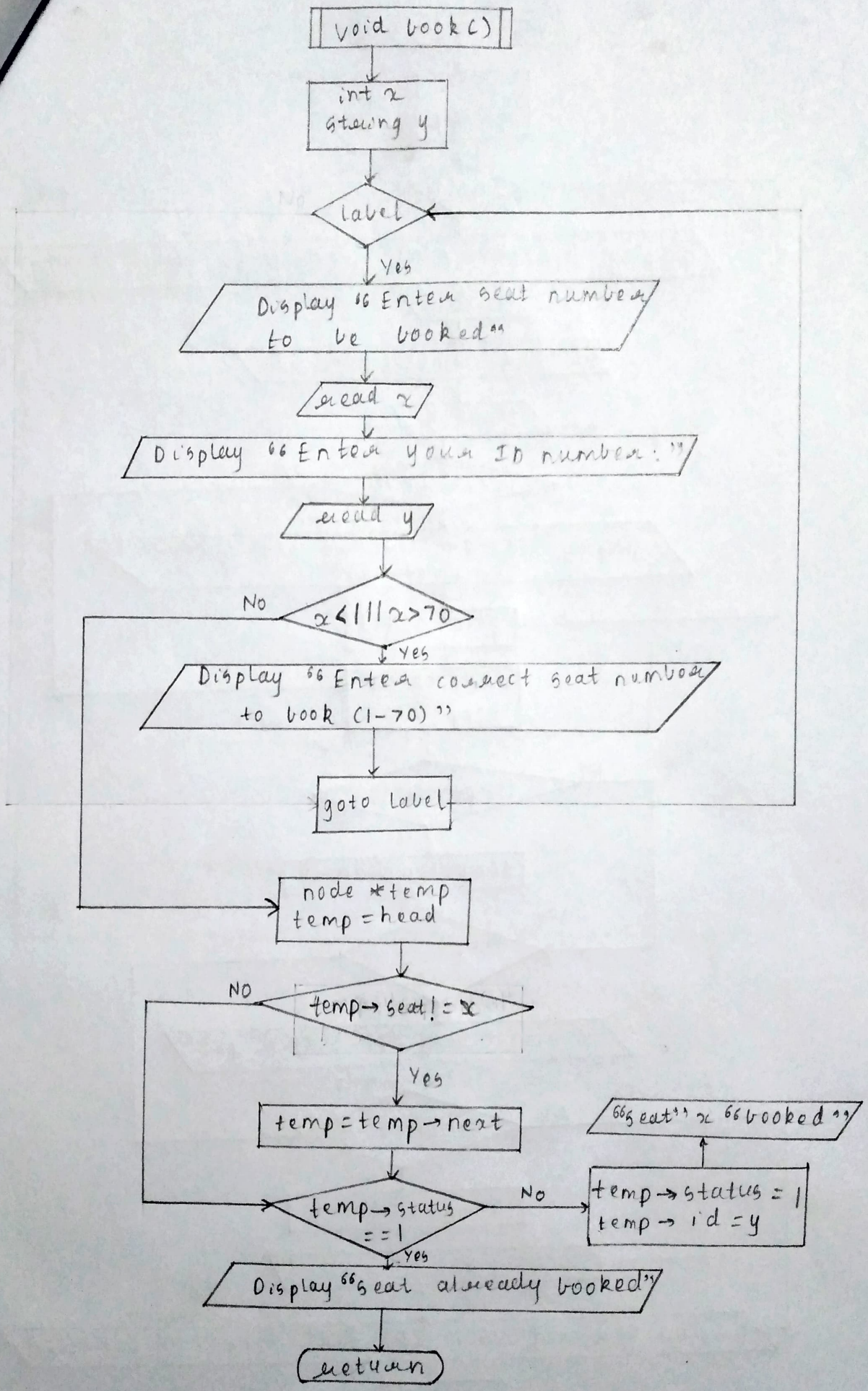
→ Flowchart for void create_list()



→Flowchart for void ~~tinema~~ display()

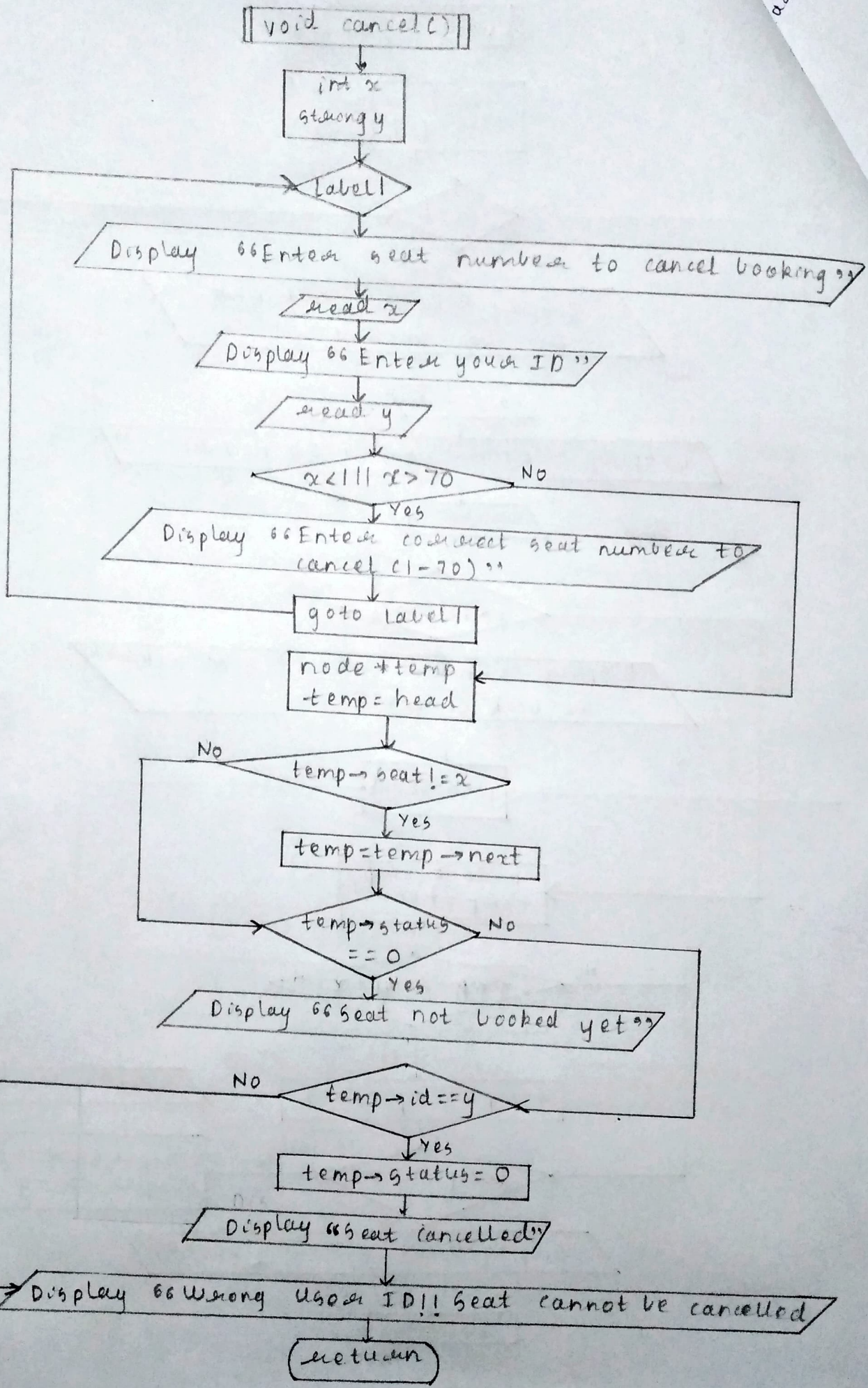


art for ~~cancel~~ void book()

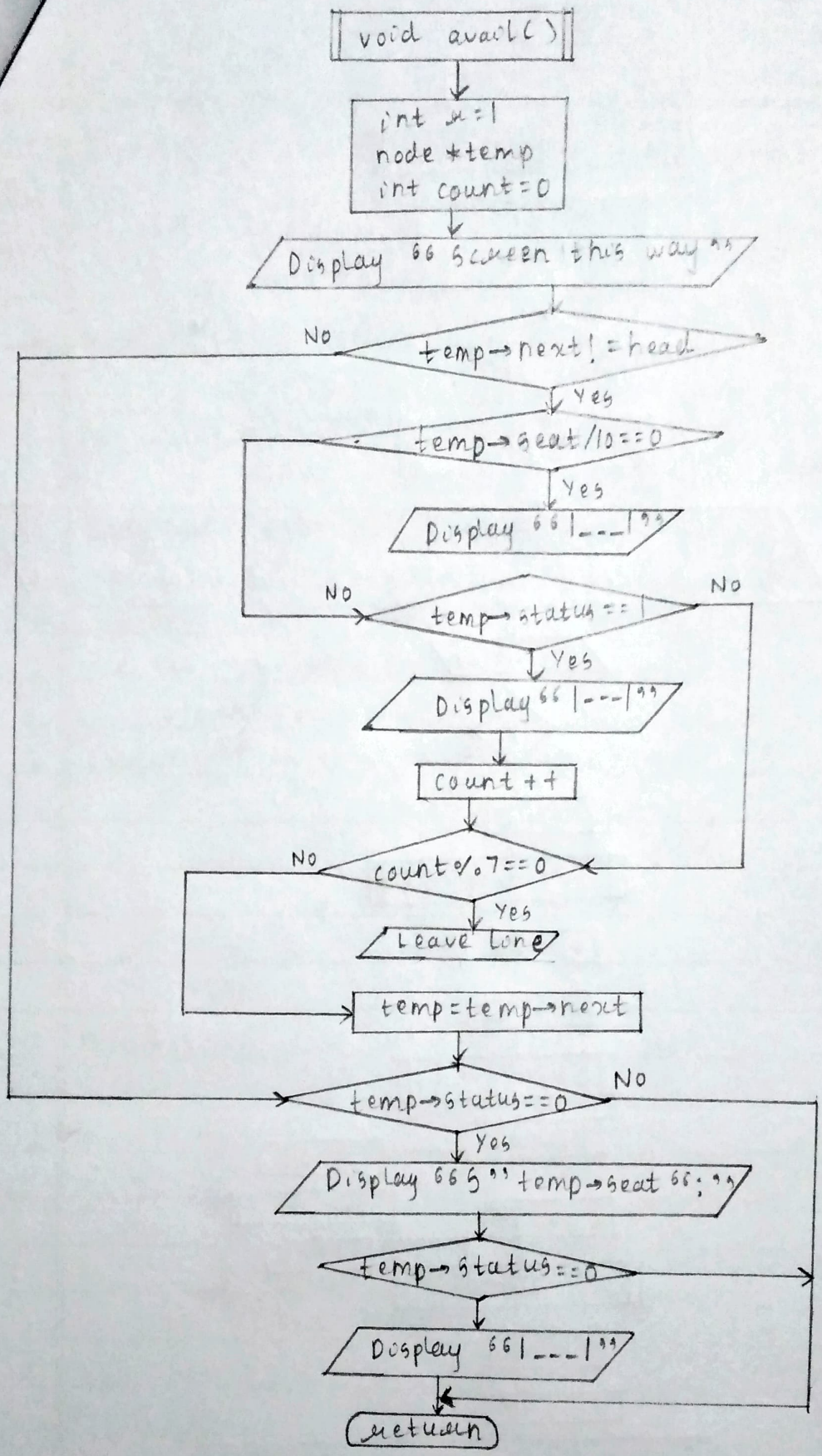


~~* Flow~~

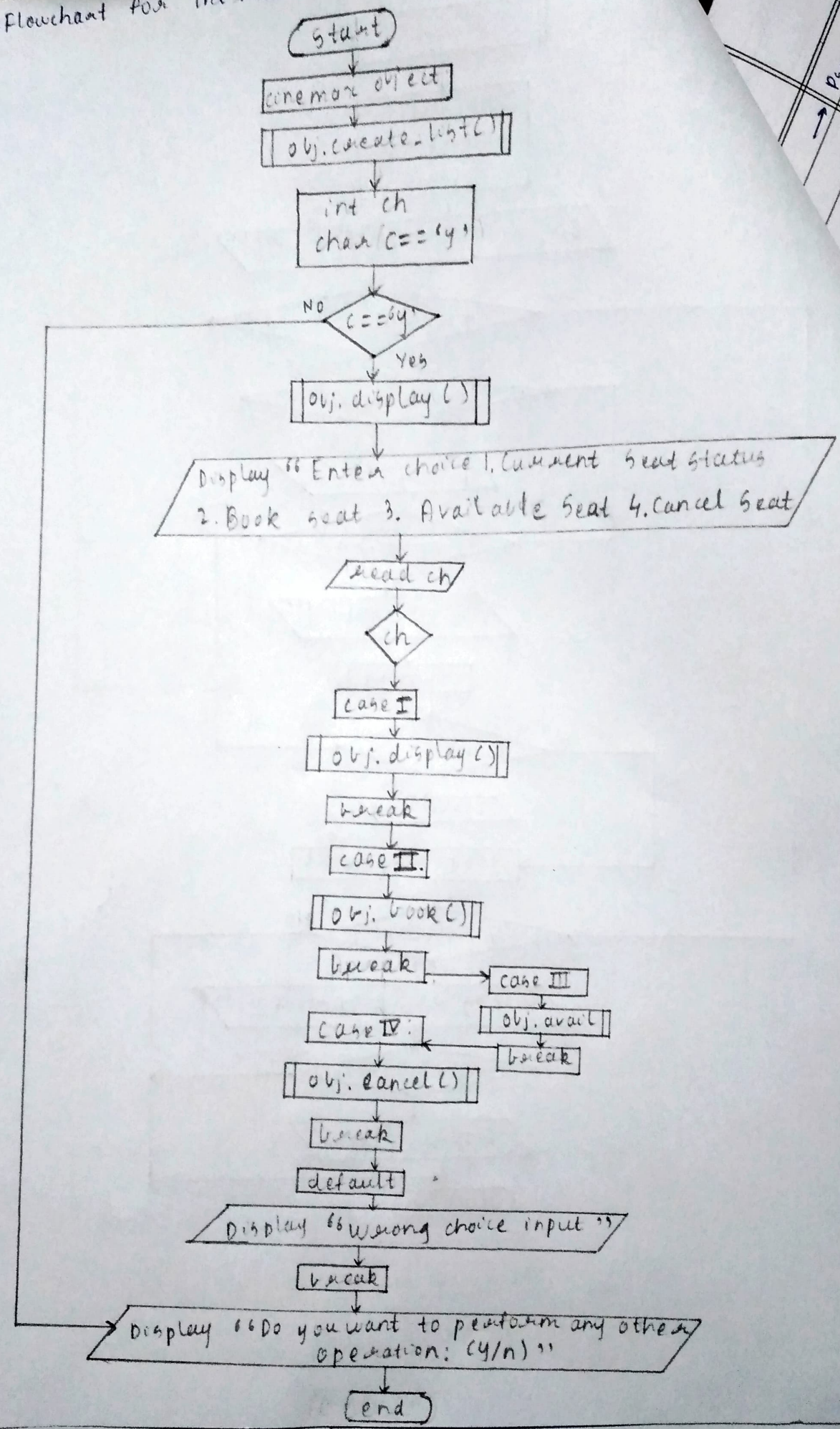
→ Flowchart for void cancel()



(art for void avail())



→ Flowchart for int main()



→ Pseudocode for class node

1. Create node* next
   node* prev
2. Declare int seat
   string id
   int status

→ Pseudo code for class cinemax

1. Create node* head, * tail, * temp
2. Create cinemax()
   store head = NULL
3. Create function void create_list()
   void display()
   void book()
   void cancel()
   void avail()

→ Pseudocode for void create_list()

1. Initialize int i=1
   temp→seat = 1
   temp→status = 0
   temp→id = "null"
   tail = head = temp
2. for (int i=2; i<=70; i++) do
   begin
   create node* p
   store p = new node

store p→seat=i

p→status=0

p→id="null"

store tail→next=p

p→prev=tail

tail=p

store tail→next=head

head→prev=tail

end

3. return

→ Pseudocode for ~~def~~ void display()

1. Initialize i=1
2. Create node* temp
3. Store temp=head
4. Initialize count=0
5. Display "Screen this way"
6. while (temp→next!=head) do

begin

if temp→status==0 then

display "|___|"

else

display "|_B_|"

increment count

if temp→seat/10==0 then

display "S0" temp→seat ":"

else

display "S" temp→seat ":"

if count %7==0 then

go to next line

increment i

end

7. Dis

```
        store temp = temp → next
    end
7.  Display "S" temp → seat ":"
8.  if temp → status == 0 then
        display "|___|"
    else
        display "|_B_|"
9.  return
```

→ Pseudocode for void book()

```
1.  Declare int x
            string y
2.  Create Label
3.  Display "Enter seat number to be booked"
4.  read x
5.  Display "Enter your ID number"
6.  read y
7.  if (x < 1 || x > 70) then
        display "Enter correct seat number to book (1-70)"
        goto Label
8.  create node *temp
9.  store temp = head
10. while temp → seat != x do
    begin
        temp = temp → next
    end
11. if temp → status == 1 then
        display "seat already booked!"
    else
```

store temp → status = 1

~~display "seat~~ already booked

store temp → id = y

display "seat " x "booked!"

12. return

→ Pseudocode for ~~de~~ void cancel()

1. Declare int x
   string y
2. Create label1
3. Display "Enter seat number to cancel booking"
4. read x
5. Display "Enter your ID"
6. ~~read~~ reading y
7. if x<1 || x>70 then
   display "Enter correct seat number to cancel (1-70)"
   goto label1
8. Create node *temp
9. store temp = head
10. while temp → seat != x do
    begin
    store temp = temp → next
    end
11. if temp → status=0 then
    display "seat not booked ~~→~~ yet!!"
12. else if ~~temp → status~~ temp → id == y then
    store temp → status = 0
    display "seat cancelled!"

else

display "Wrong User ID!!! Seat cannot be cancelled!!!

13. return

→ Pseudocode for void avail()

1. Initialize i = 1
2. Create node * temp
3. Store temp = head
4. Initialize int count = 0
5. Display "Screen this way"
6. while temp → next != head do

begin

if temp → seat/10 == 0 then

display "S0" temp → seat ":"

else

display "S" temp → seat ":"

if temp → status == 0 then

display "|___|"

else if temp → status == 1 then

display "|---|"

increment count

if count % 7 == 0 then

goto next line

temp = temp → next

end

7. if temp → status == 0 then

display "S" temp → seat ":"

8. if temp → status == 0 then

display "|___|"

9. return

→ Pseudocode for int main()

1. Start
2. Create cinemax obj
3. call function obj.create_list()
4. Declare int ch
   char c='y'
5. while c=='y' do
   begin
   display "CINEMAX MOVIE THEATRE"
   display "Enter choice 1. Current seat status
   2. Book seat 3. Available seat 4. Cancel seat"
   read ch
   switch (ch)
   case 1:
   call function obj. display()
   break
   case 2:
   call function obj.book()
   break
   case 3:
   call function obj.avail()
   break
   case 4:
   call function obj.cancel()
   break
   default:
   display "Wrong choice input"
   break

display "Do you want to perform any other operation: (y/n)"

read c

end

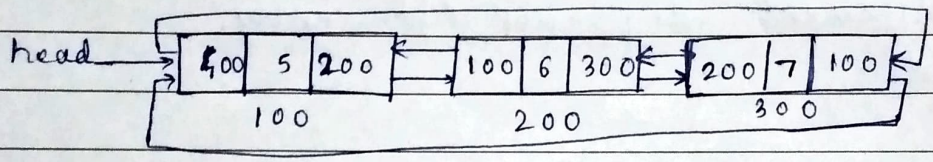
6. stop

Q1- What is a Doubly circular Linked List? Explain with example.

Ans- • ~~In doubly~~ A circular doubly linked list is a mixture of a doubly linked list and a circular linked list.

• Like the doubly linked list, it has an extra pointer called the previous ~~node~~ pointer, and similar to the circular linked list, its last node points at the head node.

→ Example:-



• As you can see, ~~the~~ each node contains the address of the previous as well as the next node.

• Similarly, the last node points to the head node.

Q2. Write applications of DCLL

Ans-1 music and video playlists:-

• Circular doubly linked lists are commonly used to implement music and video playlists, allowing users to navigate between tracks and repeat the playlist once the last track is reached.

2. Browser history:-

• Browser history can be stored using a circular doubly linked list, allowing users to navigate back and forth between previously visited pages.

3. Undo / redo functionality :-

- Many applications use circular doubly linked lists to implement undo/redo functionality, allowing users to undo or redo previous actions in the application.

4. Deque data structure :-

- A deque (double ended queue) is a data structure that allows insertion and deletion from both ends.
- Circular doubly linked lists can be used to implement dequeues efficiently.